gilles.hunault@univ-angers.fr Angers, octobre 2004

Un petit tuteur PHP

Table des matières

1. Présentation de PHP

PHP est un langage de programmation pour les serveurs *Web*, orienté base de données. Ce qui signifie qu'il permet de gérer des pages *HTML* dynamiques et en particulier de construire des pages *Web* correspondant à des résultats de requêtes sur des bases SQL. Selon ses auteurs, sa syntaxe est proche du *C*, de *Java* et de *Perl*. PHP se compose d'un langage de programmation classique (affectations, boucles...), de fonctions *Web* et de fonctions base de données. Nous décrivons ici l'utilisation de la version 3 de PHP, désignée aussi sous la référence PHP3. Ce tuteur est largement inspiré de la documentation officielle.

PHP est disponible sous *Windows* et sous *Unix*. Le site officiel est

`http://www.php.net`

Pour utiliser PHP, une fois le logiciel installé, configuré il suffit d'écrire des fichiers de type `.php3` plutôt que `.htm` et le tour est joué. L'appel à PHP est commencé par le marqueur `<?` et se termine par le marqueur `?>`. Il est à noter que dés le premier appel à PHP, le serveur *Web* gère une "session" PHP qui garde en mémoire un certain nombre d'informations, comme par exemple les variables PHP.

Ainsi le fichier suivant, nommé xmp01.php3

```
<html>
  <head>
     <title> (gH) : un premier exemple en PHP</title>
  </head>
  <body>
     Bonjour, <? echo " tout le monde " ;?>
  </body>
</html>
```

est une page *Web* qui utilise le mode normal pour afficher "Bonjour" et qui fait appel à PHP pour afficher "tout le monde".

Attention ! l'imbrication des marqueurs Web classiques et ceux de PHP n'a pas besoin d'être "équilibrée" tant que cela donne un page correcte.

Par exemple, le fichier xmp02.php3 qui contient

```
<html>
  <head>
     <title> (gH) : deuxième exemple en PHP</title>
  </head>
  <body>
    <ul>
      <li> Un
      <? echo "<li>Deux " ;
         echo "</ul>" ;
      ?>
  </body>
</html>
```

est correct, alors que le marqueur `<?` commence avant que le marqueur `</ul>` termine la liste (ce qui est normal, ici, puisque c'est PHP qui termine la liste).

On peut ouvrir et fermer PHP plusieurs fois dans une même page, comme

```
<html><head>
<? echo " <title>troisième exemple</title> " ; ?></head>
<body><ul><li> Un
<? echo "<li>Deux " ;
   echo "</ul>" ;
?>
</body></html>
```

PHP est une aide à la sécurité car c'est le serveur qui interprète la page. L'utilisateur ne voit que la page produite. Ainsi, la demande de l'affichage du texte source de la précédente donne

```
<html>
  <head>
     <title>troisième exemple</title>
  </head>
  <body>
     <li> Un
     <li>Deux
  </body>
</html>
```

comme si toute le texte avait été écrit "normalement".

2. Eléments du langage de PHP

2.1 Commentaires, variables, affectations etc.

Un commentaire peut être délimité par `/*` et `*/` ou suivre le symbole `#` ou le symbole double `//`.

Une variable est repérée par le symbole `$` et l'affectation se fait avec `=`. Par exemple `$jmois = 0` met la valeur 0 dans la variable `jmois`. PHP n'est pas typé explicitement, ce qui signifie qu'on n'a pas besoin de préciser le type d'une variable. En interne, PHP utilise les types `integer`, `float`, `string`, `array` et `object`.

Les variables chaines de caractères utilisent les deux jeux de délimiteurs `'` et `"` ainsi que les séquences *escape* comme `\n`, `\$`... Lorsque le délimiteur `"` est utilisé, les variables à l'intérieur sont "interpolées" c'est à dire remplacées par leur valeur. Ainsi

```
$i = 2 ;
echo " \$i vaut  $i " ;
```
affiche `$i vaut 2`

Les instructions PHP doivent être séparées par des points-virgules.

Un tableau en PHP est une variable qui utilise des crochets avec une valeur (indice) entre les crochets. L'indice peut être quelconque, ce qui fournit à la fois les tableaux classiques (indice numérique) et les "hashs" ou tableaux associatifs (indice chaine de caractères). Ainsi `$code["pierre"] = 12 ;` met 12 dans le tableau `code` à l'indice `pierre`.

La fonction `count()` renvoie le nombre d'éléments dans le tableau et on accède aux différents éléments du tableau avec `each()` ou à partir de `next()` et `prev()`. Les tableaux peuvent être triés avec `sort()`, `asort()`, `arsort()`, `ksort()`...

Certains tableaux sont créés dynamiquement. Ainsi `explode(char,chaine)` renvoie un tableau où chaque élément correspondant au découpage de la chaine, `char` étant le séparateur. Par exemple

```
explode(" ; "," il pleut ; je mange ; tu dors ") ;
```

renvoie un tableau a trois éléments dont le premier (situé à l'indice zéro) est `il pleut`. Si on utilise plus d'un indice, on met des crochets autour de chaque indice. Ainsi l'élément traditionnel $M_{i,j}$ d'une matrice se note `$M[$i][$j]` en PHP.

En ce qui concerne les variables, il faut noter que toute variable d'un formulaire produit une variable de même nom pour PHP. Par exemple le formulaire

```
<form action=...>
Id : <input type="text"   name="nom"><br>
     <input type="submit" value="envoi">
</form>
```

renvoie automatiquement une variable $nom qui contient ce que l'utilisateur a entré dans le champ texte correspondant.

Enfin, les variables d'environnement sont considérées comme des variables PHP normales, même si la fonction `getenv()` donne accès aux variables systèmes.

Lors de l'affectation, PHP permet de nombreux raccourcis *à la C*, comme `$a = ++$b =$c;` ce que nous ne conseillons pas pour des questions de lisibilité. L'opérateur `%` renvoie le reste de la division entière (ou "modulo"). La concaténation de chaine se fait avec un . et `$a = $a . $b` peut se raccourcir en `$a .= $b` . Les opérateurs logiques ont les "doux" noms `and`, `or`, `xor`, `!` (not) et les opérateurs de comparaison sont standards : `==`, `<=`, `>=`.

Pour les habitués au *C* et à *Perl*, il est possible d'utiliser la construction `$a ? $b : \$c ;` comme affection en `if $a then $b else $c` classique.

2.2 Tests et Boucles

Le test en *Si* se fait avec `if (condition) { ... }`. On peut utiliser `else` pour l'alternative inverse et `elseif` pour des alternatives multiples. L'instruction `switch` est également disponible pour les structures de cas générales.

La boucle *Tantque* utilise `while (condition) { ... }` tandis que la forme `do {...} while (condition)` réalise la boucle *répéter... jusqu'à* puisqu'elle effectue le test en fin de cours de boucle.

La boucle *pour* classique en PHP est `for (debut;fin;increment) { ...}`.

2.3 Sous-programmes

Un sous-programme en PHP est une forcément une fonction, déclarée par `function Nom (liste d'Arguments) { ... ; } ;` .

La valeur renvoyée éventuellement doit être spécifiée par `return`. Pour envoyer des valeurs multiples, il faut soit convertir le tout en chaine de caractères, soit renvoyer un tableau explicite comme `return array ($i, $j)` .

Les arguments d'une fonction sont facultatifs. Le passage par référence plutôt que par valeur se fait en rajoutant `&` soit dans l'en-tête de la fonction soit au moment de l'appel. Si l'argument est suivi de `= valeur` il s'agit d'une valeur constante, qui sera prise par défaut.

3. Programmation classique en PHP

3.1 Exemple 1 : un Bonjour complet

Rappelons que ce premier programme doit afficher le mot "Bonjour" puis demander à l'utilisateur son prénm et lui dire au revoir, en affichant la date, l'heure et le prénom en majuscules. L'affichage du mot Bonjour et la saisie du prénom ne relèvent pas vraiment de PHP, c'est le rôle d'un formulaire HTML. Une version minimale pour cette saisie est

```
<html>
     <head>
        <title> (gH) : programme BONJOUR en PHP, partie 1 :
                       formulaire de saisie du prénom </title>
     </head>
     <body>
     <form action="bonjour.php3">
     <font color="red">Bonjour</font>.  
     Quel est votre nom ? <input type="text" name="prenom">
     <input type="submit" value=" envoi ! ">
     </form>
     </body>
   </html>
```

Lorsque l'utilisateur clique sur le bouton "envoi !", la machine transmet l'URL `http://.../bonjour.php3?prenom=XXX` où `XXX` correspond au texte entré dans le champ du prénom (ou `http://.../bonjour.php3?prenom=` si le champ est vide). Signalons qu'il est d'ailleurs possible d'entrer directement cette URL "à la main", ce qui permet de tester le programme même sans formulaire, possibilité que nous exploiterons avec le prochain exemple.

Un programme PHP de réponse minimal est

```
<html>
      <head>
         <title> (gH) : programme BONJOUR en PHP,
                        partie 2 : réponse </title>
      </head>
      <body>
      <?
         if ($prenom=="") { $nom = " bel inconnu " ; }
                     else { $nom = $prenom  ; } ;
         $ladate  = date("d/m/Y") ;
         $lheure  = date("h:i") ;
         $nomMaj  = strtoupper($nom) ;
         echo " Le $ladate à $lheure, au revoir, $nomMaj." ;
      ?>
      </body>
    </html>
```

mais on préférera sans doute une version plus habillée, avec des `<font...>`, ce qui est plus du ressort d'HTML que de PHP.

3.2 Exemple 2 : Table de multiplication

D'habitude, le programme table de multiplication vient demander un nombre (ou utiliser la valeur passée en paramètre). Ici nous ne ferons pas de formulaire, car nous utiliserons la valeur passée sur la ligne de l'URL (avec un rappel du passage de paramètre si l'URL est entrée telle quelle).

```
<html>
<head>
   <title> (gH) : programme TABLE (version 1) en PHP </title>
</head>
<?
function aide() {

echo "pour voir la table de N, écrivez au bout de l'URL<br>";
echo "le symbole ? suivi de n=N. Ainsi pour la table de 5,<br>";
echo "à la suite de http://.../table.php3 il faut mettre ?n=5.";

} ; # fin de fonction aide
```

```
function tdm($vn) {
    echo " <h1>Table de multiplication de $vn</h1> " ;
    echo "<xmp>" ;
    for ($i=1;$i<=10;$i++) {
        $fi = sprintf("   %3d ",$i) ;
        $fp = sprintf("   %6d ",$i*$vn) ;
        echo "   $fi  fois   $vn   =   $fp\n" ;
    } ; # fin de pour i
    echo " </xmp> " ;
} ; # fin de fonction aide

?>
<body>
<?
   if ($n=="") { aide()  ; } else { tdm($n) ; } ;
?>
</body>
</html>
```

Il est possible de passer plusieurs paramètres, par exemple pour gérer la largeur de l'affichage. Le passage de paramètres, commencé par le symbole ? se prolonge par le symbole & suivi d'une expression `variable=valeur`. C'est ce qui est rappelé dans la fonction aide(). Nous profitons alors du passage de paramètre par défaut de PHP dans l'appel de la fonction tdm.

```
    <html>
      <head>
         <title> (gH) : exemple MOYENNE en PHP </title>
      </head>

      <?  function aide() { ?>

 pour voir la table de N, écrivez au bout de l'URL<br>
 le symbole ? suivi de n=N. Ainsi pour la table de 5,<br>
 à la suite de http://.../table.php3 il faut mettre ?n=5.<p>

 vous pouvez aussi passer le param\'{e}tre largeur d'affichage ;
 en utilisant & pour séparer les param\'{e}tres.<br>
 par exemple http://.../table.php3?n=123456789&largeur=15

      <?   } ; # fin de fonction aide
```

```
function tdm($vn,$larg=6) {
    echo " <h1>Table de multiplication de $vn</h1> " ;
    echo "<xmp>" ;
    for ($i=1;$i<=10;$i++) {
        $fi = sprintf("   %3d ",$i) ;
        $fp = sprintf("   %$larg"."d",$i*$vn) ;
        echo "   $fi  fois $vn   =   $fp\n" ;
    } ; # fin de pour i
    echo " </xmp> " ;
} ; # fin de fonction aide

?>
<body>
<?
   if     ($n=="")        { aide()  ; }
   elseif ($largeur=="") { tdm($n) ; }
   else                  { tdm($n,$largeur) ; } ;
?>
</body>
</html>
```

3.3 Exemple 3 : Calcul de moyennes de classe

Il s'agit ici de calculer la moyenne de notes lues dans un fichier. Nous allons en fait proposer soit d'entrer les noms et notes dans une zone texte soit d'envoyer le fichier. Nous n'utiliserons qu'un seul fichier nommé `moyenne.php3` qui s'appellera lui-même avec divers paramètres. Le programme principal vient seulement choisir le sous-programme à exécuter en fonction du paramétre `mode`, soit le texte :

```
if     ($mode=="")                   { saisie()  ;}
elseif ($mode=="saisieFormulaire") { saisieForm();  }
elseif ($mode=="calculFormulaire") { gereCalculs(1,$valeur);}
elseif ($mode=="saisieFichier")    { saisieFic();}
elseif ($mode=="calculFichier")    { gereCalculs(2,"");}
else                                { pbMode();} ;
```

La saisie est un simple formulaire qui vient relancer le script `moyenne.php3` avec le bon paramètre, soit:

```
 function saisie() {
     debutPageHtml(1," Moyenne, partie 1 : saisie ") ;
?>   <h1>Donnez votre choix :</h1>     <ul>
     <li><a href="moyenne.php3?mode=saisieFormulaire">
         calcul sur texte du formulaire</a><p>
     <li><a href="moyenne.php3?mode=saisieFichier">
         calcul sur fichier-texte</a>
     </ul></body></html>
<?   } ; # fin de fonction saisie
```

où `debutPageHtml()` est une fonction qui gère le document (avec les *tags* `<html>`, `<body>` etc.). Chaque saisie particulière utilise son formulaire `HTML` :

```
function saisieForm() {
     debutPagehtml(2," Moyenne, partie 2.1 :
                 saisie sur formulaire " ) ;
?>   <h1>Saisie des valeurs par formulaire</h1>
     <form action="moyenne.php3">
     <input type=hidden name="mode" value="calculFormulaire">
     <textarea name="valeur" rows=10 cols=35>
          Durand H. * 9 14 12
     </textarea>
     <input type="submit" value ="envoi"></form></body></html>
<?   } ; # fin de fonction saisieForm

function saisieFic() { global $nomfic, $nomfic_name ;
     debutPageHtml(3," Moyenne, partie 2.2 :
                    calcul sur fichier " ) ;
?>   <form enctype=multipart/form-data action=moyenne.php3
                                      methode=post>
     <input type=hidden   name="mode" value="calculFichier">
     <input type=hidden name=max_file_size value=1000>
     <font size=4>
     Indiquez votre fichier  <input type=file name=nomfic>
     <input type=submit value=envoi></form>
     </font>
     </body></html>
<?   } ; # fin de fonction saisieFich
```

La gestion des calculs consiste à appeler le sous-programme `calculs` avec comme paramètre une chaine de caractères qui contient toutes les valeurs entrées. Dans le cas d'une saisie dans le formulaire, on dispose tout de suite de cette variable. Dans le cas de la saisie sur fichier local, on vient lire le fichier et transférer son contenu dans cette variable. Nous avions convenu pour le format d'entrée de séparer le nom des notes par une étoile (mais pas d'imposer un nombre de notes); La partie `calculs` vient lire la variable caractère par caractère, remplaçant au passage les sauts de ligne par des espaces et découpe en mots la variable, en affectant là où il faut dès que l'étoile est trouvée. Pour les affichages, nous faisons appel à la même fonction puisque nous trions les tableaux *in situ* définies comme des variables globales.

```
<? ## moyenne.php3 (gH) 2000 ; gilles.hunault@univ-angers.fr

function debutPageHtml($num,$titre) {
    echo "<html><head><title>" ;
    echo "$titre" ;
    echo "</title></head><p><body background=beige.jpg>\n" ;
    $fint = explode(" ","red blue blue") ;
    echo "<font color=$fint[$num] size=+4>\n" ;
} ; # fin de fonction finTitre()

function afficheNomsNotes($modaf) {

   global $nomElv, $notElv, $moyElv, $nbl, $mpl ;

    $ie = 1 ;
    echo "<blockquote><table>" ;
    while ($ie<=$nbl) {
      echo "<tr><td><font size=6 color=black><b>$nomElv[$ie]
                                               </font></td>" ;
      if ($modaf==0) {
           echo " <td><font size=5 color=#800000> notes
                                               </font></td>" ;
           $in =  1 ;
           while ($in<=$nN) {
             $n = $notElv[$ie][$in]
             echo "<td align=right><font size=5>$n</td>" ;
             $in++; } ; # fin de tant que
      } ; # fin de si sur modaf
```

```
      $m =  $moyElv[$ie] ;
      $fm = sprintf("%6.2f",$m) ;
      echo "<td  align=right><font size=5 color=#800000>";
      echo "   moyenne   </td>" ;
      echo "<td  align=right><font size=6 color=green>" ;
      echo "<b><kbd>$fm</kbd></td>" ;
      $ie++ ;
      echo "</tr>" ;
    } ; # fin de tant que
    echo " </table></blockquote><p>" ;
} ; # fin de fonction afficheNomsNotes

function tri_Alpha($nE,$nN) {
    global $nomElv, $notElv, $moyElv  ;
    $ie = 1 ;
    while ($ie<=$nE-1) {
      $je = 1 + $ie ;
      while ($je<=$nE) {
         $m1 = $nomElv[$ie] ;
         $m2 = $nomElv[$je] ;
         if ($m1>$m2) {  # permutations
            $nomElv[$ie] = $m2 ; $nomElv[$je] = $m1 ;
            $tmp         = $moyElv[$ie] ;
            $moyElv[$ie] = $moyElv[$je] ;
            $moyElv[$je] = $tmp ;
         } ; # fin de tant il faut permuter
         $je++ ; } ; # fin de tant que sur $je
      $ie++ ; } ; # fin de tant que sur $ie
} ; # fin de fonction tri_Alpha($nE,$nN) {

function tri_Moyenne($nE,$nN) {
    global $nomElv, $notElv, $moyElv ;
    $ie = 1 ;
    while ($ie<=$nE-1) {
      $je = 1 + $ie ;
      while ($je<=$nE) {
         $m1 = $moyElv[$ie] ; $m2 = $moyElv[$je] ;
         if ($m1<$m2) {
            # permutations (pas <= à cause des ex-aequo)
            $moyElv[$ie] = $m2 ;  $moyElv[$je] = $m1 ;
            $tmp         = $nomElv[$ie] ;
```

```
            $nomElv[$ie] = $nomElv[$je] ;
            $nomElv[$je] = $tmp ;
         } ; # fin de tant il faut permuter
         $je++ ; } ; # fin de tant que sur $je
      $ie++ ; } ; # fin de tant que sur $ie
} ; # fin de fonction tri_Moyen($nE,$nN) {

function affichages_et_FinPageHtml($nE,$nN) {

    echo "<h1>Affichage  historique des $nE él\`{e}ve(s) avec
                           sans doute $nN note(s)</h1>" ;
    afficheNomsNotes(0) ;

    echo "<h1>Affichage par ordre alphabétique</h1>" ;
    tri_Alpha($nE,$nN) ;
    afficheNomsNotes(1) ;

    echo "<h1>Affichage par ordre de mérite</h1>" ;
    tri_Moyenne($nE,$nN) ;
    afficheNomsNotes(1) ;

    # puis on termine la page Html

    echo "</body></html>" ;

} ; # fin de fonction affichages_et_FinPageHtml

  function saisie() {

     debutPageHtml(1," Prog. Moyenne, partie 1 : saisie ") ;

?>   <h1>Donnez votre choix :</h1>
     <ul>
     <li><a href="moyenne.php3?mode=saisieFormulaire">
          calcul sur texte du formulaire</a><p>
     <li><a href="moyenne.php3?mode=saisieFichier">
          calcul sur fichier-texte</a>
     </ul>
     </body></html>

<?   } ; # fin de fonction saisie
```

```
function saisieForm() {

     $m = "Prog. Moyenne, partie 2.1 : saisie sur formulaire";
     debutPagehtml(2,$m) ;
?>   <h1>Saisie des valeurs par formulaire</h1>
     <form action="moyenne.php3">
     <input type=hidden   name="mode" value="calculFormulaire">
     <textarea name="valeur" rows=10 cols=35>
          Durand H. * 9 14 12
          Dupon P. * 18 15 11
          Aymar J. * 11 11 11
     </textarea>
     <input type="submit" value ="envoi"></form></body></html>
<?   } ; # fin de fonction saisieForm

function saisieFic() { global $nomfic, $nomfic_name ;

     $m = "Programme Moyenne, partie 2 : calcul sur fichier " ;
     debutPageHtml(3,$m) ;
?>   <form enctype=multipart/form-data action=moyenne.php3
                                         methode=post>
     <input type=hidden   name="mode" value="calculFichier">
     <input type=hidden name=max_file_size value=1000>
     <font size=4>
     Indiquez votre fichier  <input type=file name=nomfic>
     <input type=submit value=envoi></form>
     </font>
     </body></html>

<?   } ; # fin de fonction saisieFich

function calculs($valr) {

   global $nomElv, $notElv, $moyElv, $nbl, $mpl ;

   $valr .= chr(10) ;       # pour forcer la derni\`{e}re ligne
   $le    = strlen($valr) ; # longueur de la chaine
   $nbl   = 0 ;             # nombre d'él\`{e}ves
   $mot   = "" ;            # mot courant
   $ic    = 0  ;            # indice du caract\`{e}re courant
```

```
   # format d'entrée : identification  * notes, par exemple
   #  Dupont J. * 12 18

   while ($ic<$le) {

       $lec = substr($valr,$ic,1) ;
       $cec = ord($lec) ;

       if ($cec==42) { # c'est l'étoile

          $nbl++ ; $nomElv[$nbl] = trim($mot) ;
          $lec = "" ; $mot = "" ;

       } elseif ($cec==10) { $lignot = $mot ; # retour charriot

          $tlignot = explode(" ",$lignot) ;
          $cnt = count($tlignot) ;
          $im = 0 ; $nn = 0 ;

          while ($im<$cnt) {
           $motn = trim($tlignot[$im]) ;
           if (strlen($motn)>0) {
              $nn++ ; $notElv[$nbl][$nn] = $motn ;
              $moyElv[$nbl] += $motn ;
            } ; # fin de si strlen($motn)>0
            $im++ ;
          } ; # fin de tant que $im<$cn

          if ($nbl==1) {
              $mpl = $nn ;
          } ; # fin de si $nbl==1
          $moyElv[$nbl] /= $mpl ; $mot  = "" ;
       } elseif ($cec==13) { $lec = "" ;
       } ; # fin de si on voit un retour chariot

       $mot .= $lec ; $ic++ ;

   } ; # fin tant que ic < le

} ; # fin proc function calculs
```

```
function gereCalculs($lemode,$lavar) {

   global $nomElv, $notElv, $moyElv, $nbl,
          $mpl, $nomfic, $nomfic_name ;

   if     ($lemode==1) { $valeurs = $lavar ;

          debutPageHtml(1+$lemode,
             " Programme Moyenne, partie 2.2 :
               calcul sur formulaire " ) ; }

   elseif ($lemode==2) {

          debutPageHtml(3,
             " Programme Moyenne, partie 2 :
               calcul sur fichier " ) ;

          # on transf\`{e}re tout le fichier dans la variable valr

          echo "<font color=black size=4>
               Moyenne des valeurs du fichier
               $nomfic_name<br></font>" ;
          echo '<font color="#800000"blue>
              (fichier '.$nomfic.' sur serveur)</font> ' ;

          $fh  = fopen($nomfic,"r") ;
          $valeurs = "" ;  $nbl = 0 ;
          while (!feof($fh)) {
              $nbl++ ; $lig   = fgets($fh,4096) ;
              $valeurs .= $lig ;
          } ; # fin tant que non fin de fichier

          fclose($fh) ;

   } ; # fin de si sur lemode

   calculs($valeurs) ; affichages_et_FinPageHtml($nbl,$mpl) ;

} ; # fin de fonction gereCalculs
```

```
function pbMode() {

   debutPagHtml(0," Programme Moyenne, erreur ! " ) ;
   echo "</body></html>" ;

} ; # fin de fonction pbMode

################################### programme principal

if     ($mode=="")                   { saisie();}
elseif ($mode=="saisieFormulaire") { saisieForm();}
elseif ($mode=="calculFormulaire") { gereCalculs(1,$valeur);}
elseif ($mode=="saisieFichier")    { saisieFic();}
elseif ($mode=="calculFichier")    { gereCalculs(2,"");}
else                                 { pbMode();};

?>
```

4. Exemples spécifiques en PHP

4.1 Nombre de fichiers dans une répertoire

PHP dipose de fonctions systèmes. Par exemple la fonction `dir()` renvoie une variable (une *instance d'objet* en fait) de type répertoire. On peut donc lui appliquer la *méthode* `read()` et la *méthode* `close()`. Comme son nom l'indique, `read()` permet de lire les entrées du répertoire, y compris . et .. (sous *Unix*). Plus précisément, `read()` lit au premier appel la premier entrée du répertoire, puis l'entrée courante dans le répertoire à partir de de la précédente pour les appels suivants, le pointeur de position étant concervé en interne par PHP.

Pour compter le nombre de fichiers dans un répertoire, il suffit d'appeler la fonction `dir()` en lui passant comme paramètre le nom du répertoire à consulter puis à exécuter une boucle sur l'appel de la fonction `read()`. Par exemple, sur notre système, nous avons un répertoire Php/ et qui contient deux sous répertoires, Documentation/ et Programmes/. Nous avons donc écrit une fonction `dirstat()` et il suffit de lui passer en paramètre le nom d'un répertoire pour obtenir son nombre de fichiers.

Ainsi le code suivant

```
<? function dirstat($nomdir) {

# calcule le nb de fichiers dans le répertoire

$d  = dir($nomdir) ;
$nbf = 0 ;
 while ($entry=$d->read()) {
    $nbf = $nbf +1 ;
 } ; # fin tant que
 $d->close() ;
 return $nbf-2 ;

} # fin fonction dirstat

$dirbase = "/home/Gh/Info/Cours/Php/" ;
echo "<font size=+5>Comptages sur les fichiers
            du répertoire $dirbase</h1> " ;

$cdir = $dirbase ;
$nbf = dirstat($cdir) ;
echo   "<font size=+5>racine : $nbf fichiers<p>" ;

$ldir = "Documentation/" ;
$cdir = $dirbase.$ldir ;
$nbf  = dirstat($cdir) ;
echo   "<font size=+5>$ldir : $nbf  fichiers<p>" ;

$ldir = "Programmes/" ;
$cdir = $dirbase.$ldir ;
$nbf  = dirstat($cdir) ;
echo   "<font size=+5>$ldir : $nbf   fichiers" ; ?>
```

pemet d'obtenir de réaliser un comptage et un affichage qui ressemble à :

```
    Comptages sur les fichiers
       du répertoire /home/Gh/Info/Cours/Php/
    racine : 30 fichiers
    Documentation/ : 17 fichiers
    Programmes/ : 11 fichiers
```

4.2 Requêtes MYSQL simples

PHP a été conçu pou interfacer des bases de données. En plus d'un langage de programmation, c'est donc aussi un langage de requêtes. PHP version 3 sait dialoguer avec

`Adabas D`	`InterBase`	`Solid`
`dBase`	`mSQL`	`Sybase`
`Empress`	`MySQL`	`Velocis`
`FilePro`	`Oracle`	`Unix dbm`
`Informix`	`PostgreSQL`	

Une base de données regroupes plusieurs tables de données. Par exemple pour une application dont la base est nommée GESTION, les tables pourront s'appeler CLIENTS, FOURNISSEURS, ARTICLES et COMMANDES. On suppose que le lecteur, la lectrice de ce cours ont quelques connaissances sur les bases de données en général. Sinon, nous recommandons le cours de P. RIGAUX et l'ensemble du site Web correspondant qui s'adresse aux étudiants du CNAM, cycle A à l'adresse

`http://sikkim.cnam.fr/~rigaux/bdpi.html`

Nous alllons présenter ici l'accès aux bases de données SQL du gestionnaire MYSQL. MYSQL se compose d'un langage de définition de données et de droits ainsi qu'un langage de manipulation des données. MYSQL est disponible sous *Windows* et sous *Unix*. De nombreux sites *Web* gèrent leurs bases de données avec le couple `PHP/MYSQL`. Le site officiel de `MYSQL` est `http://www.mysql.com`. La version utilisée pour ce tuteur est 3.22.29.

Notre premier exemple de requête a pour but de montrer comment on établit une connection avec le serveur de bases de données. La lecture du manuel de PHP indique que la connection se fait par `mysql_connect`, que la sélection de la base de données est réalisée par `mysql_select_db`, qu'une requête utilise la syntaxe `mysql_query()` et qu'enfin on ferme la connection par `mysql_close()`.

Nous allons donc essayer d'ouvrir la base EE pour l'utilisateur YY sur le site XXX.XXX.XX.XX pour ensuite compter le nombre d'enregistrements de la table UU (pour raison de confidentialité et pour éviter de nombreux accès à nos bases, nous avons remplacé les noms de champs, les numéros IP etc. par des valeurs plus anonymes).

Le programme PHP suivant est suffisant pour réaliser le comptage :

```
<?  mysql_connect("XXX.XXX.XX.XX","YY","");
    mysql_select_db("EE");

    $res  = mysql_query("select count(*) from UU") ;
    $ligr = mysql_fetch_array($res) ;
    $nbst = $ligr["count(*)"] ;

    echo " il y a  $nbst enregistrement(s)." ;
?>
```

Détaillons un peu la gestion de la requête : à la fin de l'exécution de la fonction `mysql_query`, MYSQL renvoie vrai ($TRUE$) si la requête s'est bien passée, faux ($FALSE$) sinon. La fonction `mysql_fetch_array` permet de récupérer ($FETCH$) les différentes lignes du tableau ($ARRAY$) du résultat de la requête. Comme ce tableau est associatif, il faut utiliser l'indice qui correspond au corps de la requête. Il est à noter que la requête ne doit pas se terminer par un point-virgule alors qu'en session MYSQL ou en programme MYSQL le point-virgule est obligatoire pour terminer une instruction. Les autres instructions d'interfaçage de MYSQL disponibles en PHP3 avec leur description en anglais sont :

```
mysql_affected_rows
     Get number of affected rows in previous MySQL operation
mysql_close
     Close MySQL connection
mysql_connect
     Open a connection to a MySQL Server
mysql_create_db
     Create a MySQL database
mysql_data_seek
     Move internal result pointer
mysql_db_query
     Send an MySQL query to MySQL
mysql_drop_db
     Drop (delete) a MySQL database
mysql_errno
     Returns the number of the error from previous operation
mysql_error
     Returns the text of the error from previous operation
```

mysql_fetch_array
: Fetch a result row as an associative array

mysql_fetch_field
: Get column information from a result and return object

mysql_fetch_lengths
: Get the length of each output in a result

mysql_fetch_object
: Fetch a result row as an object

mysql_fetch_row
: Get a result row as an enumerated array

mysql_field_name
: Get the name of the specified field in a result

mysql_field_seek
: Set result pointer to a specified field offset

mysql_field_table
: Get name of the table the specified field is in

mysql_field_type
: Get the type of the specified field in a result

mysql_field_flags
: Get the flags for the specified field in result

mysql_field_len
: Returns the length of the specified field

mysql_free_result
: Free result memory

mysql_insert_id
: Get the id generated from the previous INSERT operation

mysql_list_fields
: List MySQL result fields

mysql_list_dbs
: List databases available on on MySQL server

mysql_list_tables
: List tables in a MySQL database

mysql_num_fields
: Get number of fields in result

mysql_num_rows
: Get number of rows in result

mysql_pconnect
: Open a persistent connection to a MySQL Server

mysql_query
: Send an SQL query to MySQL

```
mysql_result
     Get result data
mysql_select_db
     Select a MySQL database
mysql_tablename
     Get table name of field
```

4.3 Requêtes PHP/MYSQL plus sophistiquées

Montrons sur un exemple plus détaillé comment gérer via PHP les sorties MYSQL. Supposons que notre table que nous nommerons `diremail` de la base, disons BD, contienne des adresses-mail comme xyz@tagada.com dans le champ `email` et que l'on dispose de l'année, du mois et du jour de création de l'adresse par notre système informatique, mis respectivement dans les champs `anneR`, `moisR` et `jourR`. On voudrait avoir un rapport sur le nombre d'adresses crées. Notre premier programme vient afficher le nombre total d'adresses dans la base et liste le nombre d'adresse par mois. Par exemple on pourrait obtenir:

```
  Statistiques par mois sur les adresses :

     le 20/01/2000 à 03:42
     on a 32753 adresses

  Choisissez le mois à visualiser

     2000    Février         12     0 %
             Janvier      10400    32 %
     1999    Décembre     10478    32 %
             Novembre     11811    36 %
             Septembre       56     0 %
```

où chaque mois est cliquable. Ainsi, si on clique sur le mot Février, on obtient

```
  Statistiques par jour pour le mois de :  Février 2000

     Jour   1  :   8  Adresses
            2  :   1  Adresses
            3  :   3  Adresses
```

Notre programme commence bien sûr par établir une connection puis affiche le nombre total d'adresses. On effectue ensuite le recensement des années par un `select distinct(anneR)` que l'on trie par `order by anneR`. Une boucle `while` qui utilise `my_sql_fetch` permet de transférer les années dans un tableau PHP. La même technique permet de construire un tableau des mois. On boucle ensuite sur les années et sur les mois. Pour une année et un mois donnée, s'il y a un nombre d'adresses non nul, on construit l'ancre `<a href=...>` correspondante. La page `statmois.php3` est alors appelée qui donne le détail jour par jour suivant le même principe.

```
<? # gestion d'adresses email dans une base mysql via php

# connection SQL et ouverture de la base

mysql_connect("localhost","root","");
mysql_select_db("test");

# 1. a comptage global

$res  = mysql_query("select count(email) from BD") ;
$ligr = mysql_fetch_array($res) ;
$nbst = $ligr["count(email)"] ;

# 1. b affichage du comptage global

$dthr = date(" d/m/Y à H:i ") ;
echo " <font size=6>    " ;
echo " le $dthr " ;
echo " <br><font size=6>    " ;
echo " on a : " ;
echo " <font size=+4 color=000088><b>$nbst</b></font>adresses" ;

# 2. détail mois par mois et proposons
#    (avec sélection possible d'un mois donné)

# 2.a recensement des années

$qry  = "select distinct anneR from BD order by anneR " ;
$res  = mysql_query($qry) ;
```

```
$nbans  = 0 ;
while ($ligr=mysql_fetch_array($res)) {
  $nbans++ ;   $ans[$nbans] = $ligr["anneR"] ;
} ; # fin de tant que

echo "<h1>choisissez le mois : </h1>         " ;

# 2.b recensement des mois

$qry  = "select distinct(moisR) from BD order by moisR" ;
$res  = mysql_query($qry) ;
$nbmois = 0 ;
while ($ligr=mysql_fetch_array($res)) {
  $nbmois++ ;   $mois[$nbmois] = $ligr["moisR"] ;
} ; # fin de tant que

# 3. passage en revue des années et de mois
# on crée une ancre <a href=...> si pour le mois
# et l'année courante il y a un nombre non nul
# d'adresses

echo "<table> " ;
$ian = $nbans ;
while ($ian>=1) {
   $van = $ans[$ian] ;

   $imois = $nbmois ;
   $jmois = 0 ;
   while ($imois>=1) {
     $vmois = $mois[$imois] ;
     $qry   = "select count(email) from BD " ;
     $qry  .= 'where (moisR=".$vmois." and anneR=".$van.")' ;
     $res   = mysql_query($qry) ;
     $ligr  = mysql_fetch_array($res) ;
     $nbpa  = $ligr["count(email)"] ;

     if ($nbpa>0) { # il faut préparer l'URL
        $jmois++ ;
        $nbpa  = sprintf("%3d",$nbpa) ;
        $pct   = sprintf("%3d",round(100.0*$nbpa/$nbst)) ;
        $nmois = nomMois($vmois) ;
```

```
        echo   "<tr><td><font size=+3>  " ;
        if ($jmois==1) { echo $van ; } ;
        echo   "  </td><td><font size=+3>" ;
        $choix = $van."*".$vmois."*" ;
        echo   "<b><a href=statmois.php3?choix=$choix>
                                $nmois</a></td>" ;
        echo   "<td align=right><font size=+4 color=880000>
                             $nbpa</td>" ;
        echo   "<td align=right><font size=+4 color=008800>
                          $pct %</td></tr>" ;
     } ; # finsi sur $nbpa>0
     $imois-- ;
   } ; # fin de tant que sur $imois

  $ian-- ;
} ; # fin de tant que sur $ian
echo "</table> " ;
## -- fin de programme
?>
```

5. Après PHP 3 et MYSQ 2.2

Notre tuteur PHP3 a été écrit en janvier 2000. `PHP` et `MYSQL` n'ont cessé d'évolué depuis. L'ajout que vous lisez actuellement date d'octobre 2004 et `PHP` en est désormais à la version `5.0.2` alors que `MYSQ` est rendu à la version `5.0.1`.

`PHP` est devenu un standard pour la gestion des formulaires et des bases de données sur le *Web*. L'ajout le plus marquant pour le langage est la notion de *session* pour éviter les collisions entre variables. Quant à `MYSQL`, il s'est aussi étoffé. Nous conseillons donc de consulter les sections *What's new...* sur les sites officiels correspondants, à savoir :

```
http://www.php.net
http://www.mysql.com
```